UNIVERSIDADE PAULISTA – unip

**Ciência da computação**

**Victor Hugo de Carvalho Finotto**

**Luis Mario Carvalho Caldeira**

**ANÁLISE COMPARATIVA ENTRE BLAZOR WEBASSEMBLY E JAVASCRIPT TRADICIONAL NO DESENVOLVIMENTO WEB**

**São José do Rio Preto - SP**

**2025**

**UNIVERSIDADE PAULISTA – unip**

**Ciência da computação**

**Victor Hugo de Carvalho Finotto**

**Luis Mario Carvalho Caldeira**

**ANÁLISE COMPARATIVA ENTRE BLAZOR WEBASSEMBLY E JAVASCRIPT TRADICIONAL NO DESENVOLVIMENTO WEB**

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade Paulista – UNIP, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Me. Fabio Renato de Almeida

**São José do Rio Preto - SP**

**2025**

Espaço destinado a ficha catalográfica.

Para solicitá-la acesse o menu Serviços > Biblioteca > Ficha catalográfica.

**Victor Hugo de Carvalho Finotto**

**Luis Mario Carvalho Caldeira**

**ANÁLISE COMPARATIVA ENTRE BLAZOR WEBASSEMBLY E JAVASCRIPT TRADICIONAL NO DESENVOLVIMENTO WEB**

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade Paulista – UNIP, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Me. Fabio Renato de Almeida

Aprovado(a) em: \_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. ou Profa. Dr(a)./Me(a). xxxxxxxxxxxx

Universidade Paulista - UNIP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. ou Profa. Dr(a)./ Me(a). xxxxxxxxxxxx

Universidade Paulista - UNIP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. ou Profa. Dr(a)./ Me(a). xxxxxxxxxxxx

Universidade Paulista - UNIP

Folha opcional onde o autor presta a DEDICATÓRIA de seu trabalho. Esse elemento não recebe título e nem indicativo numérico. Elemento pré-textual complementar.

agradecimentos

Espaço opcional, o autor pode agradecer pessoas que contribuíram para elaboração de seu trabalho. Não recebe indicativo numérico. Elemento pré-textual complementar, não obrigatório.

*“Escolher a tecnologia certa não é sobre qual é a mais popular, mas qual* atende melhor ao problema.”

(*Jeff Atwood)*

resumo

Na contemporaneidade, o cenário tecnológico tem sido marcado por uma busca incessante de novas tecnologias e ferramentas que atendam às crescentes demandas do desenvolvimento. Nesse contexto, as linguagens de programação, especialmente no desenvolvimento *web*, têm ganhado cada vez mais notoriedade e, com isso, instaurado a busca por soluções que equilibrem performance e eficiência. O *JavaScript*, consolidado como linguagem predominante no desenvolvimento *web*, continua sendo amplamente utilizado devido à sua versatilidade e vasto ecossistema, compatível com diversas tecnologias e plataformas. No entanto, com o advento de novas abordagens, como o *Blazor WebAssembly*, surge uma nova perspectiva na web para os desenvolvedores. Levando isso em consideração, este estudo tem como objetivo avaliar e comparar o *JavaScript* e o *Blazor WebAssembly*, em termos de desempenho e eficiência. Esse estudo oferece uma visão abrangente das capacidades e limitações de cada linguagem, facilitando a escolha mais apropriada para cada contexto no desenvolvimento *web*, especialmente no que se refere às necessidades de performance, tempo de resposta e consumo de recursos.

**Palavras-chave:** *Blazor* *WebAssembly*; *JavaScript*; desempenho; consumo de recursos; desenvolvimento *web*;

ABSTRACT

In contemporary times, the technological landscape has been marked by a relentless pursuit of new technologies and tools capable of meeting the growing demands of software development. In this context, programming languages—particularly in web development—have gained increasing prominence, leading to the search for solutions that balance performance and efficiency. JavaScript, established as the predominant language in web development, remains widely used due to its versatility and extensive ecosystem, which is compatible with various technologies and platforms. However, with the emergence of new approaches such as Blazor WebAssembly, a new perspective arises for web developers. Considering this scenario, this study aims to evaluate and compare JavaScript and Blazor WebAssembly in terms of performance and efficiency. This research provides a comprehensive view of the capabilities and limitations of each language, assisting in the decision-making process regarding the most appropriate choice for different web development contexts, especially with respect to performance requirements, response time, and resource consumption.

**Keywords:** Blazor WebAssembly; JavaScript; performance; resource consumption; web development.

Lista de ILUSTRAÇÕES (opcional)

|  |  |  |
| --- | --- | --- |
| Figura 1 – | Título da Figura.............................................................................. | 11 |
| Foto 1 – | Título da Foto................................................................................. | 19 |
| Quadro 1 – | Título do Quadro............................................................................ | 23 |
| Foto 2 – | Título da Foto................................................................................. | 39 |

LISTA DE figuras (opcional)

|  |  |  |
| --- | --- | --- |
| Figura 1 – | Título da Figura............................................................................. | 11 |
| Figura 2 – | Título da Figura............................................................................. | 12 |
| Figura 3 – | Título da Figura. Alinhar o título da figura abaixo da primeira palavra do título quanto este apresentar mais de uma linha como neste exemplo…........……………………………………....... | 22 |

Lista de gráficos (opcional)

|  |  |  |
| --- | --- | --- |
| Gráfico 1 – | Título do Gráfico.......................................................................... | 15 |
| Gráfico 2 – | Título do Gráfico. Alinhar o título do gráfico abaixo da primeira palavra do título quanto este apresentar mais de uma linha como neste exemplo................……………….............................. | 18 |
| Gráfico 3 – | Título do Gráfico...........…........................................................... | 20 |

Lista de quadros (opcional)

|  |  |  |
| --- | --- | --- |
| Quadro 1 – | Título do Quadro. Alinhar o título do quadro abaixo da primeira palavra do título quanto este apresentar mais de uma linha, como neste exemplo..….............................................................. | 15 |
| Quadro 2 – | Título do Quadro................…...................................................... | 18 |
| Quadro 3 – | Título do Quadro..................….................................................... | 20 |

Lista de Tabelas (opcional)

|  |  |  |
| --- | --- | --- |
| Tabela 1 – | Título da Tabela............................................................................... | 23 |
| Tabela 2 – | Título da Tabela............................................................................... | 30 |
| Tabela 3 – | Título da Tabela............................................................................... | 32 |

Lista de abreviaturas (opcional)

ABREVIATURA Nome por extenso da abreviatura e em ordem alfabética

ABREVIATURA Nome por extenso

ABREVIATURA Nome por extenso

ABREVIATURA Nome por extenso

ABREVIATURA Nome por extenso

Lista de siglas (opcional)

SIGLA Nome por extenso da Sigla e em ordem alfabética

SIGLA Nome por extenso

SIGLA Nome por extenso

SIGLA Nome por extenso

Lista de SÍMBOLOS (opcional)

SÍMBOLO Nome por extenso do símbolo e em ordem alfabética

SÍMBOLO Nome por extenso

SÍMBOLO Nome por extenso

SÍMBOLO Nome por extenso

SUMÁRIO

|  |  |  |
| --- | --- | --- |
| **1** | **INTRODUÇÃO**...................................................................................... | 11 |
| **2** | **REVISÃO DE LITERATURA**................................................................ | 14 |
| **2.1** | **Título da seção secundária**................................................................ | 16 |
| 2.2.1 | Título da seção terciária..................................................................….. | 18 |
| *2.2.1.1* | *Título da seção quaternária.............................................................….* | 19 |
| **3** | **MATERIAL E MÉTODOS**..................................................................... | 21 |
| **4** | **RESULTADOS**..................................................................................... | 22 |
| **5** | **CONSIDERAÇÕES FINAIS**.................................................................. | 23 |
|  | **REFERÊNCIAS**.................................................................................... | 24 |
|  | **APÊNDICE A – TÍTULO DO APÊNDICE**............................................. | 25 |
|  | **ANEXO A – TÍTULO DO ANEXO**......................................................... | 29 |

**1. INTRODUÇÃO**

A constante evolução do cenário tecnológico, especialmente na área de desenvolvimento *web*, tem resultado na busca frequente de novas tecnologias e linguagens de programação com o objetivo de suprir a demanda por soluções mais ágeis, eficientes e performáticas. Em meio à vasta gama de opções, as decisões sobre quais tecnologias adotar se tornaram um fator determinante para assegurar a qualidade da experiência do usuário, a escalabilidade e eficiência dos projetos.

Reconhecido como o padrão para aplicações no lado do cliente (*client-side*), o *JavaScript* se firmou, ao longo do tempo, beneficiando-se de um vasto ecossistema, ampla aceitação na comunidade e integração com inúmeras ferramentas e tecnologias. No entanto, o surgimento do *WebAssembly* – um formato binário para execução de código no navegador – introduziu novas possibilidades no desenvolvimento *web*. Entre as tecnologias que exploraram esse recurso desponta o *Blazor WebAssembly* (*client-side*), proposta da Microsoft que permite desenvolver aplicações web interativas utilizando a linguagem de programação C# diretamente no navegador, sem a necessidade de *JavaScript*. Esta abordagem tem despertando o interesse dos desenvolvedores, por unir a familiaridade da linguagem com a promessa de alto desempenho em aplicações *web*.

Considerando esse contexto, esse estudo propõe uma análise comparativa entre o *JavaScript* e o *Blazor WebAssembly* visando os aspectos relacionados ao desempenho, performance, tempo de resposta e consumo de recursos. Dessa forma, pretende-se contribuir para uma tomada de decisão mais apropriada para diferentes contextos no desenvolvimento *web*.

**1.1 Objetivo Geral**

O presente estudo tem como objetivo promover uma comparação entre *JavaScript* e *Blazor WebAssembly* no contexto do desenvolvimento *web client-side*, considerando fatores como desempenho, tempo de resposta e consumo de recursos.

Por meio dessa proposta, pretende-se auxiliar desenvolvedores nas tomadas de decisões quanto à adoção da tecnologia mais apropriada pra cada contexto.

**1.2 Objetivos Específicos**

Os objetivos específicos desse estudo são:

1. Desenvolver uma aplicação web em *JavaScript* e *Blazor WebAssembly*, a partir de um mesmo modelo base, visando comparar o desempenho em tarefas específicas como carregamento inicial, tempo de resposta e eventos da interface (filtros dinâmicos, ordenações e exclusões em massa de elementos da interface).
2. Utilizar ferramentas de análise de performance dos navegadores (*DevTools*) para coletar métricas sobre o uso de memória, tempo de renderização e armazenamento local, comparando o comportamento de ambas as tecnologias durante a execução do sistema.
3. Avaliar o tamanho total dos arquivos transferidos ao carregar a aplicação (*payload*), considerando seu impacto direto no tempo de carregamento, renderização inicial e experiência do usuário.
4. Analisar a eficiência do processamento local de dados (*client-side*), incluindo filtros, paginação e renderização dinâmica, observando o tempo de execução para manipulações em grandes volumes de dados.
5. Identificar os contextos mais adequados para o uso de cada tecnologia, apontando os principais pontos fortes e limitações.

**1.3 Justificativa**

A escolha deste tema se justifica pelos seguintes fatores:

1. Evolução das tecnologias *web*: O surgimento de novas abordagens para o desenvolvimento *client-side*, como o *WebAssembly*, tem desafiado o domínio tradicional do *JavaScript*, ganhando cada vez mais espaço entre os desenvolvedores. Esse cenário, demanda estudos comparativos que avaliem o real potencial dessas tecnologias emergentes.
2. Falta de estudos práticos comparativos: O *Blazor WebAssembly*, embora promissor e cada vez mais abordado no desenvolvimento *web*, ainda carece de análises detalhadas. Avaliar seu desempenho por meio de métricas como tempo de carregamento, uso de memória e tamanho de carga (*payload*), em comparação direta com o *JavaScript*, é essencial para compreender os limites e vantagens de cada tecnologia.
3. Contribuição para a comunidade técnica: Os resultados desta análise se propõem a serem um recurso útil para a escolha consciente de tecnologias *web* em projetos modernos, que pode oferecer suporte prático à tomada de decisões na comunidade de desenvolvedores, especialmente no contexto de aplicações que envolvem manipulação de um grande volume de elementos no lado do cliente.

**1.4 Metodologia**

Para a realização deste estudo comparativo entre *JavaScript* e *Blazor WebAssembly*, será adotada uma abordagem experimental e quantitativa, baseada no desenvolvimento e análise de duas aplicações *web* com funcionalidades idênticas implementadas em ambas as tecnologias. As principais etapas abordadas na metodologia serão:

**1.4.1 Levantamento de Requisitos**

* **Definição dos objetivos funcionais:** Identificar as principais funcionalidades que deverão ser implementadas nas aplicações, como carregamento e manipulação de dados em formato JSON.
* **Requisitos funcionais:** As duas versões da aplicação devem implementar, de forma idêntica como leitura e carregamento de um arquivo JSON com grande volume de dados, filtros dinâmicos com base em critérios variados, paginação e ordenação de registros e exclusão em massa de elementos da interface.
* **Requisitos não funcionais:** Tempo de carregamento inicial (*first load*), tempo de resposta nas interações do usuário, consumo de memória e armazenamento local, tamanho do *payload* transferido e eficiência na renderização e atualização dinâmica da interface.

**1.4.2 Desenvolvimento**

* **Estruturação dos projetos**: Serão criadas duas aplicações *web* com estrutura e funcionalidades idênticas, uma implementada com *JavaScript* e outra com *Blazor WebAssembly* (C#), ambas utilizando HTML e CSS.
* **Funcionalidades implementadas**: As aplicações terão como base a manipulação de um grande volume de dados em formato JSON, realizando operações como carregamento inicial, filtros dinâmicos, paginação e ordenação e exclusão em massa de elementos.

**1.4.3 Avaliação das Ferramentas e Ambiente de Teste**

* **Padronização do ambiente**: A execução das aplicações será realizada em um ambiente controlado, utilizando o mesmo navegador, com cache limpo, e máquina, garantindo isonomia nos testes e resultados.
* **Ferramentas de medição**: Serão utilizadas ferramentas de desenvolvedor do navegador (*DevTools*) para monitoramento da aplicação durante o tempo de execução.

**1.4.4 análise e Interpretação dos Resultados**

* **Organização dos dados**: Os dados coletados serão organizados em tabelas e gráficos comparativos, permitindo uma análise objetiva entre as tecnologias, destacando os pontos fortes e limitações de cada uma nas operações propostas.
* **Discussão dos resultados**: Os dados obtidos serão interpretados no contexto de desenvolvimento *web* moderno, permitindo identificar os cenários mais adequados para o uso de cada tecnologia.

**2. FUNDAMENTAÇÃO TEÓRICA E LEVANTAMENTO BIBLIOGRÁFICO**

Nesse capítulo, serão abordados conceitos essenciais que embasam os objetivos estabelecidos neste trabalho. Serão abordadas a evolução do desenvolvimento *web* e sua influência nas aplicações modernas, as principaisarquiteturas de sistemas *web – client-side* e *server-side*, bem como o conceito e funcionamento do *WebAssembly*. Além disso, também serão exploradas as características técnicas, vantagens, limitações e particularidades das tecnologias *JavaScript* e *Blazor* *WebAssembly.*

**2.1 Evolução do Desenvolvimento Web**

**2.1.1 A Web 1.0 e os primeiros navegadores**

Com o surgimento e o crescimento exponencial da internet, a vida cotidiana tornou-se cada vez mais integrada ao ambiente digital. Esse avanço trouxe consigo a necessidade de novas formas de organizar, acessar e interagir com grandes volumes de dados e informações disponíveis. Para responder a essa demanda, a arquitetura da *Web* tem sido continuamente reinventada e aprimorada (DRAPER, 2022).

A primeira etapa dessa evolução teve início em 1991 com a proposta do cientista britânico Timothy John Berners-Lee, que desenvolveu o primeiro navegador *web* e idealizou um sistema de hipertexto distribuído, conhecida como a primeira geração *Web* ou *Web* 1.0, conforme relato de Berners-Lee (1991), citado por Gan et al. (2022). Ainda segundo os autores, as páginas *web* dessa época eram essencialmente estáticas, compostas por documentos HTML simples, com pouca ou nenhuma interatividade. Os *sites* serviam principalmente como repositórios de informação, com navegação limitada a *links* e menus básicos.

**2.1.2 A transição para a Web 2.0**

Contudo, essa estrutura começou a evoluir ao longo da década de 1990, impulsionado pela necessidade de tornar a *web* mais expressiva e interativa (OLIVEIRA, 2024). Segundo Smith (2024), em 1995, a empresa Netscape introduziu o *JavaScript* em seu navegador, reconhecendo a importância de uma linguagem de *script* capaz de dinamizar a interação nas páginas *web*. Inicialmente chamada de *LiveScript*, a linguagem foi renomeada para *JavaScript* e, desde então, se consolidou como o principal mecanismo de *scripts* no lado do cliente. Paralelamente, em 1996, foi publicada a primeira especificação oficial das folhas de estilo em cascata (CSS1), proposta por Håkon Wium Lie e desenvolvida em colaboração com Bert Bos, com o objetivo de separar conteúdo e apresentação, melhorando o controle visual das páginas (VOGEL et al., 2022).

Esses avanços técnicos, especialmente das tecnologias – HTML*,* CSS *e JavaScript* - prepararam o terreno para a transição à *Web* 2.0, que se consolidou a partir de 2004. O principal marco dessa nova era foi o empoderamento dos usuários, que passaram a ser produtores de conteúdo, não apenas consumidores, transformando a *web* de um simples repositório informacional em um espaço colaborativo e interativo. O uso de tecnologias como AJAX, *JavaScript*, e folhas de estilo avançadas permitiu a criação de sites dinâmicos e ricos em funcionalidades.

**2.1.3 O surgimento das Single Page Applications (SPAs)**

Esse cenário impulsionou o surgimento de um novo modelo de aplicação: as *Single Page Applications (SPAs),* em que toda a lógica de navegação e atualização da interface ocorre no lado do cliente, sem recarregamento da página. Essa mudança de paradigma intensificou a demanda por performance *client-side*, especialmente em contextos com interfaces complexas e com grande volume de dados manipulados diretamente pelo navegador (SMITH, 2024).

**2.1.4 Web 3.0, descentralização e novas demandas**

A partir dessa evolução, surgiu o conceito de *Web* 3.0, como uma resposta à necessidade de dar mais controle aos usuários sobre seus próprios dados. Essa fase busca tornar a *web* mais inteligente, personalizada e conectada, com o uso de tecnologias como inteligência artificial, aprendizado de máquina e *blockchain*, conforme apontam Shaikh, Siddiki e Pathak (2025).Neste mesmo cenário, ganha destaque a chamada *Web3*, que vai além da ideia de personalização e propõe uma internet verdadeiramente descentralizada, baseada em contratos inteligentes, criptomoedas e outras soluções distribuídas (DRAPER, 2020).

**2.1.5 A busca por desempenho no cliente-side e o surgimento do WebAssembly**

Com essa transformação, as aplicações *web* passaram a exigir cada vez mais das tecnologias *client-side*. Interfaces ricas, interações em tempo real e aplicações que rodam diretamente no navegador se tornaram comuns. Isso aumentou a cobrança por performance do lado do cliente - o que abriu espaço para novas soluções além do *JavaScript* tradicional. Nesse contexto, surgem tecnologias como o *WebAssembly*, que permitem executar código compilado no navegador com alto desempenho, como, por exemplo, o *Blazor WebAssembly*, trazendo a possibilidade de usar C# e .NET diretamente no *front-end*, com todas as vantagens de performance e estrutura que isso pode representar (SMITH, 2024).

**2.2 Arquiteturas de Aplicações Web**

Neste tópico, serão abordadas as principais arquiteturas utilizadas no desenvolvimento de aplicações *web*. Serão discutidas as arquiteturas *Server-Side* e *Client-Side*, ressaltando suas características, vantagens e desvantagens, além de um comparativo entre elas para ilustrar suas diferentes aplicações no contexto moderno da *web*.

**2.2.1 Arquitetura Server-Side**

A arquitetura *Server-Side* é um modelo tradicional no desenvolvimento de aplicações *web*, onde o servidor é responsável por processar as requisições do cliente, executar a lógica de negócios e retornar os resultados para o navegador, conforme apontam Damke e Gregorini (2022). Neste modelo, ainda segundo os autores, as páginas *web* são geradas no servidor e enviadas ao cliente em forma de HTML, CSS e JavaScript, por exemplo, sendo renderizadas diretamente no navegador do usuário. O servidor, nesse contexto, gerencia as operações de manipulação de dados e interação com bancos de dados.

Segundo Iskandar et al. (2020), a principal vantagem da arquitetura *Server-Side* está em centralizar todo o processamento. Dessa forma, como o código que executa a lógica da aplicação permanece no servidor, os dados do sistema estão mais protegidos e a manutenção do código torna-se mais simples de ser realizada. Além disso, o modelo *Server-Side* dispõe de um excelente controle sobre o fluxo de dados e as interações do usuário, uma vez que a maior parte do processamento ocorre no servidor, garantindo uma diminuição considerável na complexidade e risco de inconsistências.

Porém, as desvantagens desse modelo incluem uma dependência maior de uma boa infraestrutura de servidores, que devem ser projetados e dimensionados para garantir o desempenho em ambientes de grande escala, com um fluxo maior de interações. Ademais, como cada interação do usuário requer uma nova comunicação com o servidor, o tempo de resposta e latência podem ser afetados diretamente, impactando a experiência dos usuários, apontam Damke e Gregorini (2022).

**2.2.2 Arquitetura Client-Side**

Em contraste com a arquitetura *Server-Side*, a arquitetura *Client-Side* transfere a maior parte da responsabilidade de processamento para o lado do cliente, ou seja, para o navegador do usuário. Nesse modelo, a lógica de aplicação é executada diretamente no cliente, utilizando tecnologias como JavaScript e frameworks como React, Angular, Vue.js, entre outros (SMITH, 2024).

A principal vantagem da arquitetura *Client-Side* é a agilidade no processamento das interações do usuário (FRAJTÁK et al., 2014 apud NEVES et al., 2023). Segundo Damke e Gregorini (2022), as aplicações podem responder de forma muito mais performática, sem a necessidade de estabelecer uma constante comunicação com o servidor para cada interação, uma vez que o código é executado diretamente no navegador. Dessa forma, ainda em consonância com a ideia dos autores, isso resulta em uma experiência mais fluida e dinâmica para o usuário. Além disso, ao reduzir a carga sobre o servidor, a arquitetura *Client-Side* pode ser mais escalável em cenários de grande volume de acessos.

Por outro lado, essa abordagem exige que o navegador do cliente tenha capacidade de processar a lógica de negócios e renderizar a interface de forma eficiente (NEVES et al., 2023). Além disso, a segurança se torna um desafio, uma vez que o código é enviado para o cliente e pode ser acessado e manipulado, o que requer cuidados especiais no desenvolvimento para evitar vulnerabilidades.

**2.2.3 Comparativo entre Server-Side e Client-Side**

Ao comparar as arquiteturas *Server-Side Rendering* (SSR) e *Client-Side* *Rendering* (CSR), é possível identificar diferenças cruciais em termos de desempenho, segurança, escalabilidade e complexidade de desenvolvimento. Ambas as abordagens têm suas vantagens e limitações, e a escolha entre elas depende dos requisitos específicos do projeto.

A arquitetura *Server-Side* tende a ser mais segura, já que o código e os dados sensíveis permanecem no servidor, minimizando os riscos de exposição ao usuário. Conforme Iskandar et al. (2020), o servidor processa as requisições do usuário, executa a lógica de negócio e envia ao cliente uma página completamente renderizada, o que contribui para a proteção dos dados e da lógica interna da aplicação. No entanto, essa abordagem pode ser mais lenta em termos de desempenho, pois cada interação do usuário requer uma nova requisição ao servidor, o que pode resultar em maior latência e tempo de resposta. Além disso, seguindo a ideia dos autores, a escalabilidade em sistemas *Server-Side* depende fortemente da infraestrutura do servidor.

Em contrapartida, a arquitetura *Client-Side* se destaca pela rapidez nas interações, uma vez que o processamento é feito no próprio navegador. Isso pode melhorar a experiência do usuário, especialmente em aplicações interativas e dinâmicas. No entanto, a segurança e a compatibilidade entre navegadores tornam-se desafios a serem superados (NEVES et al., 2023). Além disso, a complexidade do desenvolvimento pode aumentar, pois o desenvolvedor precisa garantir que o código seja eficiente e funcione corretamente em diferentes dispositivos e navegadores.

A escolha entre uma arquitetura *Server-Side* ou *Client-Side* deve ser feita com base nas necessidades do projeto, levando em consideração o tipo de aplicação, os requisitos de desempenho, a segurança e a escalabilidade desejada. Esta análise comparativa é essencial para compreender as decisões de arquitetura que influenciam o desenvolvimento web moderno, especialmente em projetos que utilizam Blazor WebAssembly e JavaScript tradicional, por exemplo.

**2.3 WebAssembly em Profundidade**

Com o avanço das arquiteturas web e a popularização do modelo Client-Side, surge o WebAssembly (WASM) como uma tecnologia inovadora que amplia as possibilidades do desenvolvimento web ao permitir a execução de código compilado com alta performance em ambiente de navegador. Nesse contexto, com o objetivo de superar as limitações do JavaScript tradicional em aplicações mais complexas, o WebAssembly busca oferecer uma alternativa mais eficiente para cenários que demandam maior desempenho

* + 1. **Conceito e Funcionamento do WebAssembly**

Conforme a documentação oficial do WebAssembly (2025), o termo, resultado da combinação entre “Web”, que se refere ao ambiente de execução em navegadores e à plataforma web, e “Assembly”, que remete à linguagem de baixo nível utilizada para programação próxima ao hardware, representa um formato binário portátil que permite a execução de código compilado diretamente no navegador. Dessa forma, essa combinação reflete o objetivo do WebAssembly: permitir que linguagens compiladas, sejam de baixo ou alto nível, possam ser executadas com desempenho próximo ao nativo diretamente no navegador, expandindo as possibilidades de desenvolvimento além do tradicional uso do JavaScript.

Todavia, diferentemente do JavaScript, que é interpretado ou compilado em tempo real (*just-in-time*), o WebAssembly (WASM) é pré-compilado, reduzindo a carga de processamento durante a execução e permitindo que aplicações sejam executadas de forma mais rápida e eficiente (Wang et al., 2021). Essa abordagem tem permitido o uso de linguagens como C# no desenvolvimento web, por meio de tecnologias como o Blazor WebAssembly, que aproveitam o desempenho do WASM sem abrir mão da produtividade de linguagens de alto nível.

Por essa razão, WebAssembly se torna especialmente útil em cenários onde a eficiência computacional é crucial, como em jogos com gráficos avançados, editores de imagem e vídeo em tempo real, ou aplicações científicas e de engenharia que exigem cálculos intensivos. Além disso, por manter interoperabilidade total com o JavaScript, o WebAssembly permite que ambas as tecnologias sejam utilizadas de forma complementar, unindo a flexibilidade do JavaScript à performance otimizada do WASM (AMMAR et al., 2024).

**2.3.2 Vantagens e Limitações do WebAssembly**

[...]

**2.4 Blazor WebAssembly**

[...]

**2.4.1 Conceito e Estrutura do Blazor WebAssembly**

[...]

**2.4.2 Vantagens e Limitações do Blazor WebAssembly**

[...]

**2.5 JavaScript no Desenvolvimento Web**

**2.5.1 História e Evolução da Linguagem**

**2.5.2 Ecossistema e Ferramentas**

**2.5.3 Pontos fortes e limitações do JavaScript**

**2 REVISÃO DE LITERATURA**

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

1. texto;
2. texto;
3. texto.

Para citações diretas com mais de três linhas utilizar fonte tamanho 11, espaçamento simples e recuo de 4 cm. Norma ABNT para citação é a NBR 10520. Citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação citação (Autor, ano, página)

**2.1 Subtítulo do capítulo**

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto:

1. texto;
2. texto;
3. texto,

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Figura 1.

Figura 1 – Título da figura

Fonte: Colocar a fonte, (ano).

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Figura 2.

Figura 2 – Título da figura

Fonte: Autor (ano, p.\_\_).

Legenda: texto texto texto texto texto texto texto texto texto.

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Figura 3.

Figura 3 – Título da figura

Várias figuras pequenas:

dispor lado a lado**.**

Várias figuras pequenas:

dispor lado a lado.

Fonte: Autor (ano, p. \_\_).

2.2.1 Título da seção terciária

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto. Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto

*2.2.1.1Título da seção quaternária*

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Quadro 1.

Quadro 1 – Título do quadro

| **Texto** | **Texto** | **Texto** | **Texto** |
| --- | --- | --- | --- |
| Texto | Texto | Texto | Texto |
| Texto | Texto | Texto | Texto |
| Texto | Texto | Texto | Texto |
| Texto | Texto | Texto | Texto |

Fonte: Autor (ano, p.\_\_).

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Quadro 2 a seguir.

Quadro 2 – Título do quadro

| **Texto** | **Texto** | **Texto** |
| --- | --- | --- |
| Texto | Texto | Texto |
| Texto | Texto | Texto |
| Texto | Texto | Texto |
| Texto | Texto | Texto |

Fonte: Adaptado de Autor (ano).

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Tabela 1.

Tabela 1 – Título da tabela

| **Texto** | **Texto** | **Texto** |
| --- | --- | --- |
| Texto | 1 | 5 |
| Texto | 2 | 6 |
| Texto | 3 | 7 |
| Texto | 4 | 8 |

Fonte: Elaborado pelo autor (ano).

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Tabela 2.

Tabela 2 – Título da tabela

| **Texto** | **Texto** | **Texto** |
| --- | --- | --- |
| Texto | 1 | 10 |
| Texto | 2 | 9 |
| Texto | 3 | 8 |
| Texto | 4 | 7 |
| Texto | 5 | 6 |
| Texto | 6 | 5 |
| Texto | 7 | 4 |
| Texto | 8 | 3 |
| Texto | 9 | 2 |
| Texto | 10 | 1 |

Fonte: Elaborado pelo autor (ano).

# 3 MATERIAL E MÉTODOS

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# 4 RESULTADOS

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Gráfico 1.

Gráfico 1 – Título do gráfico



Fonte: Elaborado pelo autor (ano).

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto Gráfico 2.

Gráfico 2 – Título do gráfico



Fonte: Elaborado pelo autor (ano).

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# 5 CONSIDERAÇÕES FINAIS

Última sessão do texto onde são apresentados os desfechos finais da pesquisa, correspondentes aos objetivos ou hipóteses. É importante apresentar novas ideias, abrindo caminho a outros pesquisadores que poderão trabalhar no assunto.

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

DRAPER, C. **The Internet's Evolution: From Web 1.0 to 2.0: The Good, The Bad and The Ugly.** Journal of Digital History, 2022. Disponível em: https://www.researchgate.net/publication/360638266\_Evolution\_of\_the\_Internet\_from\_Web\_10\_to\_Metaverse\_The\_Good\_The\_Bad\_and\_The\_Ugly. Acesso em: 14 maio 2025.

GAN, Wensheng; YE, Zhenqiang; WAN, Shicheng; YU, Philip S. **Web 3.0: The Future of Internet.** Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2022. Disponível em: https://arxiv.org/pdf/2304.06032. Acesso em: 8 maio 2025.

OLIVEIRA, Flávio Donizeti de. **Desenvolvimento de sistemas web modernos**. *Revista Gestão em Foco*, ano 2024, p. 215. Disponível em: http://www.unifia.edu.br. Acesso em: 9 maio 2025.

VOGEL, Lucas; SPRINGER, Thomas; WÄHLISCH, Matthias. **Files to Streams: Revisiting Web History and Exploring Potentials for Future Prospects**. TU Dresden, 2022. Disponível em: https://arxiv.org/abs/2303.14184. Acesso em: 8 maio 2025.

DRAPER, C. **The Rise of Web 3.0: Blockchain, Decentralization, and the Future of the Internet.** Springer, 2020. Disponível em: https://link.springer.com/article/10.1007/s10586-021-03301-8. Acesso em: 14 maio 2025.

SHAIKH, Aftab; SIDDIKI, Shahin; PATHAK, Rashmi. **The Rise of Web 3.0: Decentralization and Its Impact on Web Development.** International Journal of Scientific Research and Engineering Development, v. 8, n. 2, p. 2214–2220, 2025. Disponível em: https://ijsred.com/volume8/issue2/IJSRED-V8I2P351.pdf. Acesso em: 14 maio 2025.

Smith, J**. Web Evolution to Revolution: Navigating the Future of Web Application Development**, 2024. Disponível em: https://www.researchgate.net/publication/378893182\_Web\_Evolution\_to\_Revolution\_Navigating\_the\_Future\_of\_Web\_Application\_Development. Acesso em: 14 maio 2025.

DAMKE, Gabriel Trevisan; GREGORINI, Daniel Mahl. **Comparação de Desempenho em Páginas Web: Uma Análise de Client-Side Rendering e Server-Side Rendering**. In: Congresso Latino-Americano de Software Livre e Tecnologias Abertas (Latinoware), 2022. Sociedade Brasileira de Computação. Disponível em: https://sol.sbc.org.br/index.php/latinoware/article/view/31567. Acesso em: 15 maio 2025.

ISKANDAR, Taufan Fadhilah et al. **Comparison between client-side and server-side rendering in the web development.** IOP Conference Series: Materials Science and Engineering, v. 801, p. 012136, 2020. Disponível em: https://iopscience.iop.org/article/10.1088/1757-899X/801/1/012136/pdf. Acesso em: 15 maio 2025

NEVES, Lucas Rocha; CARVALHO JUNIOR, Átila; HORA, Henrique Monteiro Rego da. **Aplicações web client-side baseadas em JavaScript: uma análise bibliométrica**. 2020. Disponível em: https://www.researchgate.net/publication/347949078\_APLICACOES\_WEB\_CLIENT-SIDE\_BASEADAS\_EM\_JAVASCRIPT\_UMA\_ANALISE\_BIBLIOMETETRICA. Acesso em: 18 maio 2025.

AMMAR, Mohammed; HUSSAIN, Faraz; MAFRA, João; MAKHLOUF, Walid. ***WebAssembly and security: A review***. Journal of Web Semantics, v. 84, p. 100776, 2024. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S157401372500005X. Acesso em: 18 maio 2025.

YAN, Yutian; TU, Tengfei; ZHAO, Lijian; ZHOU, Yuchen; WANG, Weihang. **Understanding the Performance of WebAssembly Applications**. *Proceedings of the ACM Internet Measurement Conference*, 2021. Disponível em: https://weihang-wang.github.io/papers/imc21.pdf. Acesso em: 18 maio 2025.

MEDAVARAPU, Sai Vaibhav, **Server-Side Rendering vs. Client-Side Rendering in Blazor**, Journal of Scientific and Engineering Research, 2021. Disponível em: https://jsaer.com/download/vol-8-iss-12-2021/JSAER2021-8-12-311-317.pdf

WEBASSEMBLY. What is WebAssembly? Disponível em: https://webassembly.org/. Acesso em: 18 maio 2025.

**REFERÊNCIAS (OBRIGATÓRIO)**

Modelo para Livro

SOBRENOME, Nome. **Título da obra.** Edição. Local: Editora, ano.

Modelo para E-book

SOBRENOME, Nome. **Título da obra**. Local: Editora, ano. *E-Book*. Disponível em: http://www.site.com.br. Acesso em: dia mês ano.

Modelo para Artigo de Periódico em meio eletrônico

SOBRENOME, Nome. Título do artigo: subtítulo (se houver). **Título do Periódico**, Local, volume, número, página. inicial-final, ano. Disponível em: http://www.site.com.br. Acesso em: dia mês ano.

Modelo para Teses, dissertações e trabalhos acadêmicos

SOBRENOME, Nome. **Título**: subtítulo (se houver). Orientador: Fulano de Tal. Ano. Folhas. Dissertação (Mestrado em XXX) – Universidade XXX, Local, ano.

Modelo Autor Entidade

NOME DA ENTIDADE. **Título**: subtítulo **(se houver)**. Local: Editora, ano.

**JAVASCRIPT**

https://multiversa.edu.br/docs/revista-cientifica/ARTIGO\_4\_FRAMEWORKS%20JAVASCRIPT\_ANÁLISE%20COMPARATIVA\_PEDRO\_GUESDE.pdf

<https://www.researchgate.net/profile/Henrique-Da-Hora/publication/347949078_APLICACOES_WEB_CLIENT-SIDE_BASEADAS_EM_JAVASCRIPT_UMA_ANALISE_BIBLIOMETETRICA/links/5fe9da6ba6fdccdcb80d36bf/APLICACOES-WEB-CLIENT-SIDE-BASEADAS-EM-JAVASCRIPT-UMA-ANALISE-BIBLIOMETETRICA.pdf>

https://developer.mozilla.org/en-US/docs/Web/JavaScript

**WEBASSEMBLY**

https://sbrc.sbc.org.br/2024/Ch6.pdf

**BLAZOR**

<https://learn.microsoft.com/pt-br/aspnet/core/blazor/?view=aspnetcore-8.0>

**CLIENT-SIDE E SERVER-SIDE**

<https://sol.sbc.org.br/index.php/latinoware/article/view/31567/31370>

**HISTORIA WEB**

<https://portal.unisepe.com.br/unifia/wp-content/uploads/sites/10001/2024/06/Desenvolvimento-de-Sistemas-Web-Modernos-pág-215-a-229-2.pdf>

<https://arxiv.org/pdf/2403.07828>

https://arxiv.org/pdf/2304.06032

APÊNDICE A – TÍTULO DO APÊNDICE (opcional)

Texto ou documento elaborado pelo autor, com o objetivo de

complementar o trabalho.

Formatação livre.

ANEXO A – TÍTULO DO ANEXO (opcional)

Texto ou documento não elaborado pelo autor, que serve de fundamentação para complemento do trabalho.

Formatação livre.